Universidad de las Fuerzas Armadas

“Sistema de Gestión de Proyectos de TIC

Integrantes:

1. Brayan Stehp Mendoza Márquez
2. Nicolas Aldair Gualoto Pillajo
3. Bryan Santiago Ortega Vaca
4. Luis Felipe Ibujes Franco
5. Melissa Verónica Quishpe Choloquinga

Curso:

1323

Asignatura De Programación orientada a objetos

Docente:

LUIS ENRIQUE JARAMILLO MONTAÑO

05 de diciembre de 2024

1. Idea grupal del proyecto a realizar y UMLS

Para llegar a elegir el tema de “Sistema de Gestión de Reservas de Restaurantes” pensamos en varias ideas antes que se adapten a los lineamientos dados algunas de ellas fueron un sistema financiero el cual se descarto por que pesamos que iba a ser muy complicado plasmarlo en el código y también se nos ocurrió partir con el ejemplo de la biblioteca que se vio en el control de lectura 2 pero no nos llamó la atención tampoco al final se nos ocurrió el tema que acabamos planteando ahora y a continuación explicamos cómo funciona.

Registrar Mesas:

- Cada mesa debe tener un número único, una capacidad (número de personas que puede acomodar) y un estado ("Disponible" o "Reservada").

Registrar Clientes:

- Los clientes deben tener un nombre, un número de contacto y un correo electrónico.

Crear Reservas:

- Registrar una reserva asociando a un cliente con una mesa, indicando la fecha, hora y el número de personas.
- Cambiar el estado de la mesa a "Reservada".

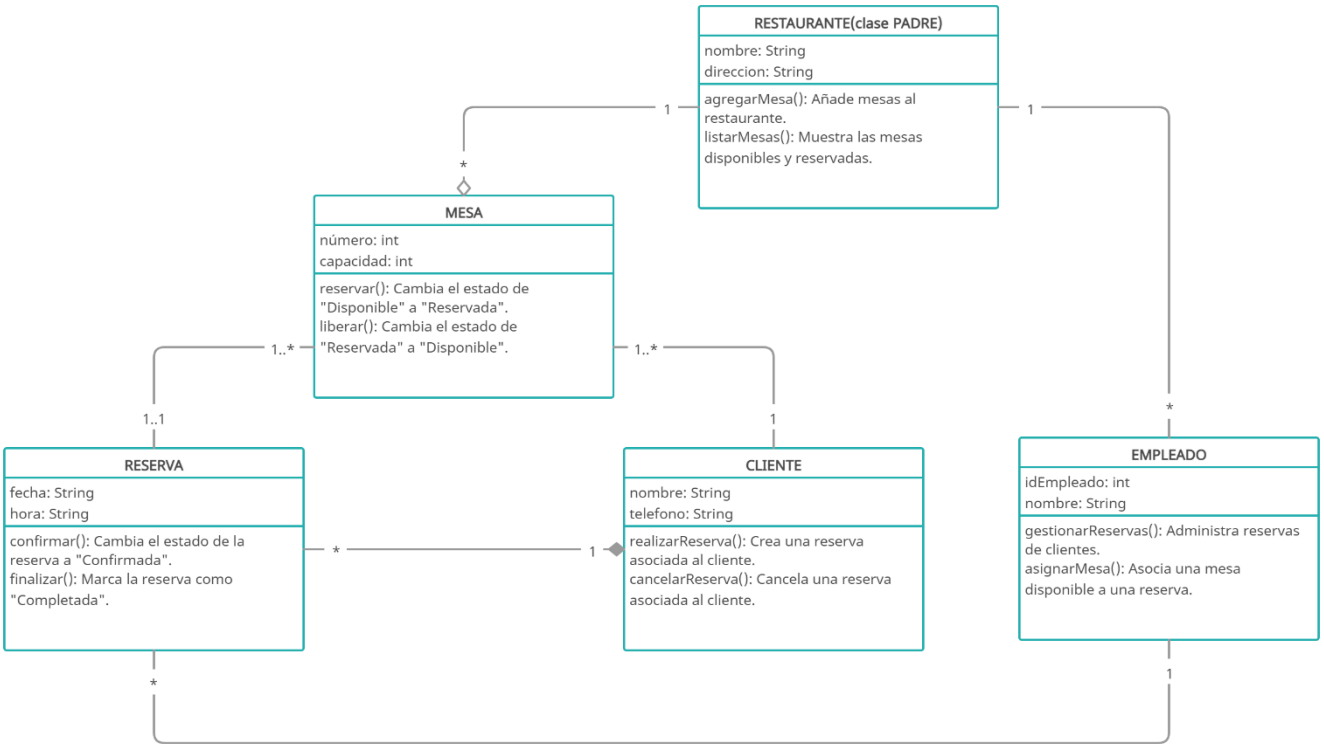
Consultar Información:

- Listar todas las mesas disponibles, mostrando su capacidad.
- Mostrar el historial de reservas, incluyendo información sobre el cliente, mesa y fecha.

Actualizar Estado de Reservas:

- Cambiar el estado de una reserva de "Pendiente" a "Completada" cuando el cliente haya utilizado la mesa.
- Liberar la mesa al cambiar su estado a "Disponible".

1.1 UMLS



2. **Relaciones**

Relaciones

Cliente ↔ Reserva

- **Relación**: Composición
- Un cliente puede realizar una o más reservas, y las reservas dependen directamente del cliente (si el cliente se elimina, sus reservas también).

Reserva ↔ Mesa

- **Relación**: Asociación
- Una reserva está asociada a una mesa específica, y una mesa puede estar asociada a una sola reserva a la vez.

Empleado ↔ Reserva

- **Relación**: Asociación
- Un empleado administra las reservas realizadas por los clientes, como gestionarlas o asignar mesas.

Restaurante ↔ Mesa

- **Relación**: Agregación
- El restaurante contiene una lista de mesas, pero las mesas pueden existir independientemente del restaurante.

Restaurante ↔ Empleado

- **Relación**: Asociación
- El restaurante tiene empleados que trabajan en la gestión de las reservas y la operación general del sistema.

3. Código

```java
import java.util.ArrayList;

// Clase Mesa
class Mesa {
    public int numero;
    public int capacidad;
    public boolean reservada;

    public Mesa(int numero, int capacidad) {
        this.numero = numero;
        this.capacidad = capacidad;
        this.reservada = false;
    }

    public void reservar() {
        if (!reservada) {
            reservada = true;
            System.out.println("Mesa " + numero + " reservada.");
        } else {
            System.out.println("La mesa " + numero + " ya está reservada.");
        }
    }

    public void liberar() {
        if (reservada) {
            reservada = false;
            System.out.println("Mesa " + numero + " liberada.");
        } else {
            System.out.println("La mesa " + numero + " ya está disponible.");
        }
    }

    public int getNumero() {
        return numero;
    }
```

```java
// Clase Cliente
class Cliente {
    public String nombre;
    public String telefono;

    public Cliente(String nombre, String telefono) {
        this.nombre = nombre;
        this.telefono = telefono;
    }

    public Reserva realizarReserva(String fecha, String hora, Mesa mesa) {
        if (!mesa.isReservada()) {
            mesa.reservar();
            return new Reserva(fecha, hora, this, mesa);
        } else {
            System.out.println("La mesa " + mesa.getNumero() + " no está disponible.");
            return null;
        }
    }

    public String getNombre() {
}
```

```java
// Clase Reserva
class Reserva {
    public String fecha;
    public String hora;
    public Cliente cliente;
    public Mesa mesa;
    public boolean confirmada;

    public Reserva(String fecha, String hora, Cliente cliente, Mesa mesa) {
        this.fecha = fecha;
        this.hora = hora;
        this.cliente = cliente;
        this.mesa = mesa;
        this.confirmada = false;
    }

    public void confirmar() {
        confirmada = true;
        System.out.println("Reserva confirmada para " + cliente.getNombre() + " en la mesa " + mesa.getNumero());
    }

    public void finalizar() {
        mesa.liberar();
        System.out.println("Reserva finalizada para " + cliente.getNombre());
    }
}

// Clase Empleado
class Empleado {
    public int idEmpleado;
    public String nombre;

    public Empleado(int idEmpleado, String nombre) {
        this.idEmpleado = idEmpleado;
        this.nombre = nombre;
    }

    public void gestionarReservas(Reserva reserva) {
        reserva.confirmar();
    }

    public void asignarMesa(Cliente cliente, Mesa mesa, String fecha, String hora) {
        cliente.realizarReserva(fecha, hora, mesa);
    }
}
// Clase Restaurante
class Restaurante {
    public String nombre;
    public String direccion;
    public ArrayList<Mesa> mesas;

    public Restaurante(String nombre, String direccion) {
        this.nombre = nombre;
        this.direccion = direccion;
        this.mesas = new ArrayList<>();
    }

    public void agregarMesa(Mesa mesa) {
        mesas.add(mesa);
        System.out.println("Mesa " + mesa.getNumero() + " agregada al restaurante.");
    }

    public void listarMesas() {
        System.out.println(x:"Mesas en el restaurante:");
        for (Mesa mesa : mesas) {
            String estado = mesa.isReservada() ? "Reservada" : "Disponible";
            System.out.println("Mesa " + mesa.getNumero() + " - Capacidad: " + mesa.capacidad + " - Estado: " + estado);
        }
    }
}
```

```java
public class restaurantepderoso {
    Run | Debug
    public static void main(String[] args) {
        // Crear restaurante
        Restaurante restaurante = new Restaurante(nombre:"El Buen Sabor", direccion:"Calle Principal 123");

        // Crear mesas
        Mesa mesa1 = new Mesa(numero:1, capacidad:4);
        Mesa mesa2 = new Mesa(numero:2, capacidad:2);
        restaurante.agregarMesa(mesa1);
        restaurante.agregarMesa(mesa2);
        // Listar mesas
        restaurante.listarMesas();
        // Crear cliente
        Cliente cliente = new Cliente(nombre:"Juan Pérez", telefono:"123456789");
        // Crear empleado
        Empleado empleado = new Empleado(idEmpleado:1, nombre:"Ana López");
        // Realizar reserva
        Reserva reserva = cliente.realizarReserva(fecha:"2024-12-10", hora:"19:00", mesa1);
        if (reserva != null) {
            empleado.gestionarReservas(reserva);
        }
        // Listar mesas después de la reserva
        restaurante.listarMesas();
        // Finalizar reserva
        if (reserva != null) {
            reserva.finalizar();
        }
        // Listar mesas después de finalizar la reserva
        restaurante.listarMesas();
    }
}
```

EJCUCION DEL CODIGO

```
PS C:\Users\RYZEN\Github>  & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.5.11-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\RYZEN\App
ata\Roaming\Code\User\workspaceStorage\d3bfbc36972b0a5123e3df3d6db7639e\redhat.java\jdt_ws\Github_3b5e9415\bin' 'AC'
Mesa 1 agregada al restaurante.
Mesa 2 agregada al restaurante.
Mesas en el restaurante:
Mesa 1 - Capacidad: 4 - Estado: Disponible
Mesa 2 - Capacidad: 2 - Estado: Disponible
Mesa 1 reservada.
Reserva confirmada para Juan Pérez en la mesa 1
Mesas en el restaurante:
Mesa 1 - Capacidad: 4 - Estado: Reservada
Mesa 2 - Capacidad: 2 - Estado: Disponible
Mesa 1 liberada.
Reserva finalizada para Juan Pérez
Mesas en el restaurante:
Mesa 1 - Capacidad: 4 - Estado: Disponible
Mesa 2 - Capacidad: 2 - Estado: Disponible
```

4. Resumen explicación final y conclusión

Desarrollamos este sistema como parte de nuestras prácticas de programación orientada a objetos. El propósito era simular un sistema de reservas para un restaurante, en el que clientes puedan realizar reservas, empleados las gestionen y el restaurante lleve un control de sus mesas. Utilizamos conceptos clave de POO como encapsulamiento, herencia y asociación entre clases.

Descripción del Proyecto

El proyecto está escrito en Java y utiliza la clase ArrayList de la biblioteca estándar para manejar la colección de mesas. Organizamos nuestro código en varias clases que representan las entidades del sistema: Mesa, Cliente, Empleado, Reserva y Restaurante.

Bibliotecas:

Utilizamos java.util.ArrayList para almacenar las mesas. Esta clase es ideal para colecciones dinámicas debido a su facilidad de uso y flexibilidad.

Recursos de Aprendizaje:

Consultamos el libro de programación orientada a objetos proporcionado por el ingeniero. Esto nos ayudó a entender los principios de diseño y las buenas prácticas.

Reforzamos conceptos con videos de YouTube, donde vimos ejemplos prácticos de la implementación de sistemas similares en Java. Estos videos nos dieron ideas para estructurar nuestro código y mejorar la interacción entre clases.

Conclusión

El proyecto nos permitió aplicar la teoría de POO en un escenario práctico. Aprendimos a colaborar en equipo, dividir responsabilidades y buscar soluciones cuando enfrentamos problemas técnicos. El sistema que desarrollamos es una base sólida y puede extenderse para incluir funcionalidades adicionales como administración de horarios, reportes de reservas y más